UNITED STATES DEPARTMENT OF COMMERCE
Patent and Trademark Office
Address: COMMISSIONER OF PATENTS AND TRADEMARKS
Washington, D.C. 20231

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.
09/026,790	02/20/98	DUPENLOUP G	30454-122(P-

LMC1/1003

MITCHELL SILBERBERG & KNUPP
11377 WEST OLYMPIC BOULEVARD
LOS ANGELES CA 90064-1683

EXAMINER	
THOMPSON, A	
ART UNIT	**PAPER NUMBER**
2768	11

DATE MAILED: 10/03/00

Please find below and/or attached an Office communication concerning this application or proceeding.

Commissioner of Patents and Trademarks

UNITED STATES PATENT AND TRADEMARK OFFICE
UNDERSECRETARY OF COMMERCE FOR INTELLECTUAL PROPERTY AND DIRECTOR OF THE UNITED STATES PATENT OFFICE
Washington, D.C. 20231

BEFORE THE BOARD OF PATENT APPEALS AND INTERFERENCES

Paper No. 11

Application Number: 09/026,790

Filing Date: February 20, 1998

Appellant: Guy Dupenloup

Joseph G. Swan
For Appellant

EXAMINER'S ANSWER

Response to Appellant's Brief On Appeal filed August 24, 2000.

(1) Real Party in Interest

A statement identifying the real party in interest as LSI Logic Corporation is contained in the Appellant's Brief on Appeal.

(2) Related Appeals and Interferences

A statement identifying the related appeals and interferences which will directly affect or be directly affected by or have a bearing on the decision in the pending appeal is contained in the brief. Appellant is unaware of any related appeals or interferences.

(3) Status of Claims

Appellant's Brief on Appeal **correctly states** the status of the claims.

(4) Status of Amendments After Final

Appellant correctly states the status of amendments after final rejection.

(5) Summary of Invention

Appellant's Summary of the Invention contained in the brief is acceptable.

(6) Issues

Appellant's concise statement of the Issues Presented on Appeal is considered correct..

(7) Grouping of Claims

The rejection of claims 1-6 and 9-20 stand or fall together because appellant's brief does not include a statement that this grouping of claims does not stand or fall together and reasons in support thereof. See 37 CFR 1.192(c)(7).

(8) Claims Appealed

Appellant's Brief Appendix contains a correct copy of the appealed claims.

(9) Prior Art of Record

Following is a listing of the prior art of record relied upon in the rejection of the claims under appeal.

5,812,416	GUPTE et al.	09/1998

(10) Grounds of Rejection

The grounds of rejection applicable to the appealed claims follows: Claims 1-6 and 9-20 are rejected under 35 U.S.C. 102(e) as being anticipated by Gupte et al. (hereinafter "Gupte"), U.S. Patent 5,812,416. An aligned comparison of the rejected claims with the prior art follows.

Claim 1

method of generating synthesis scripts to synthesize integrated circuit (IC) designs from a generic netlist description into a gate-level description;	Column 3, lines 8-10 and 22-34.
identifying hardware elements in the generic netlist;	column 14, lines 12-17 discloses parsing HDL code. Parsing inherently involves identification of hardware elements.
determining key pins for each of said identified hardware elements;	see Figure 5, step 350; column 8, lines 43-64, the inputs/outputs listed are key pins; Figure 6, steps 406, 408; column 9, lines 47-65.
extracting design structure and hierarchy from the generic netlist;	column 14, lines 12-22.
generating script to cause a logic synthesis tool to apply bottom-up synthesis to modules and sub-modules of the IC design;	column 14, lines 32-25, lines 39-48.
generating script to cause a logic synthesis tool to apply top-down characterization to modules and sub-modules of the IC design;	column 14, lines 49-52.
generating script to cause a logic synthesis tool to repeat said bottom-up and said top-down applications until constraints are satisfied.	Figure 14 illustrates the bottom up and top-down synthesization process, and especially step 812. See also, column 14, lines 52-55.

Claim 2

step of extracting design structure allows for a multilevel structuring of modules of the IC design.	Column 14, line 65 to column 16, line 24 indicates the use of various commands that impact design hierarchy.

Claim 3

generating script to cause a logic synthesis to apply initial mapping to the IC design.	Figure 12; column 13, lines 10-49; Column 14, lines 39-41.

Claim 4

the logic synthesis tool is the Synopsys Design Compiler.	Column 7 lines 6-19 discloses that one embodiment of the invention uses the Synopsys Design Compiler.

Claim 5

the step of rearranging the design hierarchy by changing the design.	Column 3, lines 8-21 teaches that incremental changes to design modules may result in hierarchical changes.

Claim 6

the step of generating script to cause a logic synthesis tool to ungroup modules of the IC design.	Figures 15B and 15C. See also column 14, lines 65-67; column 15, lines 43-67; column 16, lines 1-24. Gupte teaches the processing of a recipe file that performs an UNGROUP procedure.

Claim 9

apparatus for generating synthesis scripts to synthesize integrated circuit (IC) designs in RTL level description into gate-level description;	Figures 1,2; column 5, line 58 to column 6, line 28.
comprising a processor with connected memory; with memory having instructions for the processor;	processor is illustrated by Figure 2, block 102; column 6, lines 9-17; memory connected to processor is shown in Figure 2, block 104; column 6, lines 9-17. At column 5, lines 58-67, Gupte teaches that the computer programs (instructions) that implement the invention are stored in memory.
determining key pins for identified hardware elements from a generic netlist;	see Figure 5, step 350; column 8, lines 43-64 the inputs/outputs listed are key pins; Figure 6, steps 406, 408; column 9, lines 47-65.
extracting design structure and hierarchy from the generic netlist;	column 14, lines 12-22.
apply bottom-up synthesis to modules and sub-modules of the IC design;	column 14, lines 32-25, lines 39-48.
apply top-down characterization to modules and sub-modules of the IC design;	column 14, lines 49-52.
repeat said bottom-up and said top-down applications until constraints are satisfied;	Figure 14 illustrates the bottom up and top-down synthesization process, and especially step 812. See also, column 14, lines 52-55.
create design compile scripts to synthesize modules and sub-modules having said satisfied constraints.	Column 14, lines 39-55.

Claim 10

apparatus for generating synthesis scripts to synthesize integrated circuit (IC) designs in RTL level description into gate-level description;	Figures 1,2; column 5, line 58 to column 6, line 28.
means for determining key pins for identified hardware elements from a generic netlist;	see Figure 5, step 350; column 8, lines 43-64, the inputs/outputs listed are key pins; Figure 6, steps 406, 408; column 9, lines 47-65.
means for extracting critical design structure and hierarchy from the generic netlist;	column 14, lines 12-22.
means for applying bottom-up synthesis to modules and sub-modules of the IC design;	column 14, lines 32-25, lines 39-48.
means for applying top-down characterization to modules and sub-modules of the IC design;	column 14, lines 49-52.
means for repeating said bottom-up and said top-down applications until constraints are satisfied;	Figure 14 illustrates the bottom up and top-down synthesization process, and especially step 812. See also, column 14, lines 52-55.
means for creating design compile scripts to synthesize modules and sub-modules having said satisfied constraints.	Column 14, lines 39-55.

Claim 11

computer storage medium containing instructions for generating synthesis scripts to synthesize integrated circuit (IC) designs in RTL level description into gate-level description;	Figures 1; column 5, line 58 to column 6, line 5. Gupte discloses a floppy disk [computer storage medium] utilized to store and retrieve program code that implements the Gupte invention.
identifying hardware elements in the generic netlist;	column 14, lines 12-17 discloses parsing HDL code and parsing inherently involves identification of hardware elements.
determining key pins for identified hardware elements from a generic netlist;	see Figure 5, step 350; column 8, lines 43-64, the inputs/outputs listed are key pins; Figure 6, steps 406, 408; column 9, lines 47-65.
extracting critical design structure and hierarchy from the generic netlist;	column 14, lines 12-22.
applying bottom-up synthesis to modules and sub-modules of the IC design;	column 14, lines 32-25, lines 39-48.
applying top-down characterization to modules and sub-modules of the IC design;	column 14, lines 49-52.
repeating said bottom-up and said top-down applications until constraints are satisfied;	Figure 14 illustrates the bottom up and top-down synthesization process, and especially step 812. See also, column 14, lines 52-55.
creating design compile scripts to synthesize modules and sub-modules having said satisfied constraints.	Column 14, lines 39-55.

Claim 12

computer storage medium selected from a group consisting of magnetic device, optical device, magneto-optical device, floppy diskette, CD-ROM, magnetic tape, computer hard drive, and memory card.	At column 5, line 63 to column 6, line 2, Gupte teaches the use of other computer-readable media in addition to a floppy disk.

Claim 13

process for generating synthesis scripts to synthesize integrated circuit (IC) designs in RTL level description into gate-level description;	Figures 13, 14; column 14, lines 4-48.
identifying hardware elements in the generic netlist;	column 14, lines 12-17 discloses parsing HDL code and parsing inherently involves identification of hardware elements.
determining key pins for identified hardware elements from a generic netlist;	see Figure 5, step 350; column 8, lines 43-64, the inputs/outputs listed are key pins; Figure 6, steps 406, 408; column 9, lines 47-65.
extracting design structure and hierarchy from the generic netlist;	column 14, lines 12-22.
applying bottom-up synthesis to modules and sub-modules of the IC design;	column 14, lines 32-25, lines 39-48.
applying top-down characterization to modules and sub-modules of the IC design;	column 14, lines 49-52.
repeating said bottom-up and said top-down applications until constraints are satisfied;	Figure 14 illustrates the bottom up and top-down synthesization process, and especially step 812. See also, column 14, lines 52-55.
creating design compile scripts to synthesize modules and sub-modules having said satisfied constraints.	Column 14, lines 39-55.

Claim 14

computer system for generating synthesis scripts to synthesize integrated circuit (IC) designs in RTL level description into gate-level description;	column 13, lines 10-67 to column 14, lines 1-3.
means for determining key pins for identified hardware elements from a generic netlist;	see Figure 5, step 350; column 8, lines 43-64 the inputs/outputs listed are key pins; Figure 6, steps 406, 408; column 9, lines 47-65.
means for extracting critical design structure and hierarchy from the generic netlist;	column 14, lines 12-22.
means for applying bottom-up synthesis to modules and sub-modules of the IC design;	column 14, lines 32-25, lines 39-48.
means for applying top-down characterization to modules and sub-modules of the IC design;	column 14, lines 49-52.
means for repeating said bottom-up and said top-down applications until constraints are satisfied;	Figure 14 illustrates the bottom up and top-down synthesization process, and especially step 812. See also, column 14, lines 52-55.
means for creating design compile scripts to synthesize modules and sub-modules having said satisfied constraints.	Column 14, lines 39-55.

Claims 15-20

". . .wherein I/O conditions and constraints of the modules of the IC design captured during the top-down characterization are used to re-optimize the IC design during the bottom-up synthesis."	column 14, lines 12-55.

(11) Response to Argument

I. Introduction

In this Examiner's Answer, Examiner respectfully asserts the validity of the rejection of Appellant's Claims 1-20 under 35 U.S.C. 102(e) as being anticipated by Gupte et al., U.S. Patent 5,812,416 for at least, but not exclusively for, the following reasons:

(i) Gupte teaches the use of a generic netlist;

(ii) Gupte teaches identifying hardware elements in a generic netlist;

(iii) Gupte teaches determining key pins for the identified hardware elements;

(iv) Gupte teaches extracting design structure and hierarchy from a generic netlist.

Although Appellant partitions the claims into two (2) groups, Appellant traverses the same aforementioned reasons within both groups by reiterating similar arguments. This pattern of traversal further evinces that the claim groups are not patently distinct and the claims stand or fall together as anticipated by the singular Gupte reference.

II. The Claim Rejections Are Valid Under 35 U.S.C. 102

Appellant cites tests for determining anticipation under 35 U.S.C. 102 from In Re Bond, 15 U.S.P.Q.2d 1566, 1567 (Fed. Cir. 1990) and Richardson v. Suzuki, 868 F. 2d. 1226, 9 U.S.P.Q.2d 1913,1920 (Fed. Cir. 1989). Succinctly stated, the test for determining anticipation comprises an element-by-element equivalence between a prior art reference and an invention at bar. Appellant either misinterprets or deliberately misstates this test when he rephrases it as "Anticipation under §102 is a strict word-for-word identity test in which a single prior art

reference must show, with word-for-word identity, every element of the invention." Appellant's Brief on Appeal at 5. Neither In Re Bond or Richardson states or suggests any such word-for-word identity test for anticipation under §102.. Appellant's argument continues ". . .the above word-for-word identity tests for establishing anticipation under §102 has not been met for any of the . . .claims." Appellant's Brief on Appeal at 5. Appellant's word-for-word identity test has not been met because no such word-for-word identity test exists for determining anticipation.

Kalman v. Kimberly-Clark Corp., 713 F.2d 760, 771, 218 U.S.P.Q. 781, 789 (Fed. Cir. 1983), states the test for anticipation under 35 U.S.C. 102 this way:

> A party asserting that a patent claim is anticipated under 35 USC 102 must demonstrate, among other things, identity of invention. [O]ne who seeks such a finding must show that each element of the claim in issue is found, either expressly described or under principles of inherency, in a single prior art reference, or that the claimed invention was previously known or embodied in a single prior art device or practice.

See also Richardson v. Suzuki Motor Co., 868 F.2d 1226, 1236, 9 U.S.P.Q.2d 1913, 1920 (Fed. Cir. 1989).

The Kalman test is completely satisfied by applying the Gupte prior art reference against Appellant's Claims 1-20. No other test is required, and no court has ever approved the use of Appellant's word-for-word identity test. Therefore, Examiner maintains that the Claim rejections under 35 U.S.C. 102 using an identity of elements test as outlined by Kalman are valid.

III. Gupte Teaches A Generic Netlist

Appellant's specification defines generic netlist in Appellant's specification on page 2, lines 24-26: "A generic netlist is a netlist created from the RTL code that has not yet been correlated with a technology specific library of cells. A technology specific netlist, or a mapped netlist, is a netlist created after the design has been mapped into a particular technology-specific library of cells". Appellant's specification at page 2. Appellant's specification also includes a well-known description of RTL code: "[T]he Register Transfer Level (RTL) is typically implemented using a HDL (high level description) language". Appellant's specification at page 1, lines 16-19.

Gupte teaches that HDL is used for behavioral specification (Gupte, column 1, lines 36-38). It is well known in the art of circuit design that a behavioral specification is not technology specific. Gupte further illustrates a HDL behavioral specification that has not yet been correlated with a technology specific library of cells. When the HDL code is not technology-specific, it must be considered generic. Gupte further states in column 4, lines 55-60, that during the design phase, *arbitrary* HDL code is transformed into SBHDL code that is technology specific (emphasis added). The arbitrary HDL code is synonymous with Appellant's definition of generic netlist. Still further Gupte, in Figure 12, illustrates a high level design flow wherein block 708 "HDL code and constraints" is input to a synthesis tool (714) with Foundry technology libraries (716) and produces synthesized gate level netlists (718). The HDL code of block 708 is generic because it has not yet been correlated with the technology specific library of cells of block 716. That technology correlation step happens later in the design process in the synthesis tool block

714. Therefore, Gupte's HDL code is synonymous with Appellant's definition of generic netlist because it is not technology-specific. Now admittedly, Gupte does not use the term generic netlist. But anticipation under 35 U.S.C. 102 does not require a word-for-word equivalence as Appellant maintains. An elemental equivalence is all that is required and Gupte, accordingly, teaches the elements of Appellant's generic netlist when Gupte specifies and illustrates non-technology specific HDL code.

IV. Gupte Teaches Identifying Hardware Elements in a Generic Netlist

Appellant's specification defines the well known in the art step of identifying key hardware elements. Page 10, lines 11-12 of the specification states, "RTL code. . .can be parsed in order to identify key hardware elements." This is precisely what Gupte teaches. Gupte, at column 14, lines 12-17 discloses parsing HDL code. Parsing is a process extremely well known in the art of synthesis and it inherently involves identification of hardware elements. But even if this process were not well known, Appellant's own specification describes parsing a the process by which key hardware elements are identified. Yet, appellant not only ignores what is well-known in the art but also what is written in Appellant's own specification, and persists in the argument that parsing does not inherently involve determining key pins. No further evidence, extrinsic or otherwise, is necessary to establish the parsing element equivalence between Appellant's specification and Gupte. Appellant would be well-advised to read Appellant's own specification to clearly understand that Gupte does anticipate this limitation of Appellant's claims.

IV. Gupte Teaches Determining Key Pins for the Identified Hardware Elements

Appellant's disclosure describes the process of determining key pins on page 10, lines 13-17. "With respect to these key hardware elements, key pins with the elements' active edges or levels can also be identified." From Appellant's specification, it is apparent that the step of determining key pins is part of the step of identifying hardware elements. The key pins comprise input, output or tristate pins associated with hardware elements. Gupte teaches the use of an IOS file that involves the identification of the I/O in column 8, lines 43-64.

V. Gupte Teaches Extracting Design Structure and Hierarchy from a Generic Netlist

Gupte refers to the limitation of extracting design structure in column 14, line 12-22 which states in part, "At step 754 [of figure 13], the HDL code is parsed and the appropriate data structures are created. During script generation, the hierarchy port names and instantiations are utilized while pure behavioral code is skipped. At step 760, the system links the data structures created when the HDL code was parsed and the data structures created from the IOS file."

Appropriate data structures are created because they are extracted during the parsing of the HDL code. The data structures are not independently created. They are extracted or determined from the HDL code. Gupte's system recognizes the existence of design hierarchy and extracts information accordingly. As Gupte states in column 14, lines 23-27, "At [Figure 13] step 762, the systems flattens the design hierarchy. . .The system then. . .adds the information [e.g. hierarchy port names] to the flattened design hierarchy."

Gupte does not use the word extraction as Appellant may prefer. But anticipation under 35 U.S.C. 102 does not require word equivalence. It requires element equivalence and where, as in Gupte, the elements teach extracting design structure and hierarchy from a generic netlist are present, Gupte validly anticipates this limitation of Appellant's claims.

VI. Conclusion

Appellant's invention is anticipated in entirety by Gupte. Appellant argues that a word-for-word test is necessary for determining anticipation. However, this is not the test for anticipation under 35 U.S.C. 102 used by the courts and this should be evident to Appellant from the cases cited by Appellant in Appellant's own brief. The test for anticipation under 35 U.S.C. 102 involves an element-by-element identity correlation between the prior art and the claimed invention. Examiner has provided Appellant with an element-by-element identity correlation between Appellant's claims and Gupte for at least the following limitations: Gupte's HDL code is a generic netlist; Gupte teaches identifying hardware elements in the generic netlist and determining key pins for the identified hardware elements; Gupte teaches extracting design structure and hierarchy from a generic netlist. The relevant supporting references for these teachings and others claims by Appellant were provided not only in this Examiner's Answer but also in all prior detailed actions and responses sent to Appellant. Appellant's claimed invention is not patentably distinct over Gupte and Examiner's rejections of Claims 1 to 6 and 9 to 20 should be sustained.

Based on the above reasoning, Examiner respectfully requests that the rejections be sustained.

Respectfully submitted,

AMT
September 28, 2000

KEVIN J. TESKA
SUPERVISORY
PATENT EXAMINER

Dr. P. R. Lintz (conf.)
September 28, 2000

V. Siek (conf.)
September 28, 2000

MITCHELL SILBERBERG & KNUPP LLP
11377 West Olympic Boulevard
Los Angeles, California 90064